Ultra-Low-Latency Spiking Neural Networks with Temporal-Dependent Integrate-and-Fire Neuron Model for Objects Detection

Chengjun Zhang [1] **Yuhao Zhang** [1] **Jie Yang** [2] **Mohamad Sawan** [1 2]

Abstract

Spiking Neural Networks (SNNs), inspired by the brain, are characterized by minimal power consumption and swift inference capabilities on neuromorphic hardware, and have been widely applied to various visual perception tasks. Current ANN-SNN conversion methods have achieved excellent results in classification tasks with ultra-low time-steps, but their performance in visual detection tasks remains suboptimal. In this paper, we propose a delay-spike approach to mitigate the issue of residual membrane potential caused by heterogeneous spiking patterns. Furthermore, we propose a novel temporal-dependent Integrate-and-Fire (tdIF) neuron architecture for SNNs. This enables Integrate-and-fire (IF) neurons to dynamically adjust their accumulation and firing behaviors based on the temporal order of time-steps. Our method enables spikes to exhibit distinct temporal properties, rather than relying solely on frequency-based representations. Moreover, the tdIF neuron maintains energy consumption on par with traditional IF neuron. We demonstrate that our method achieves more precise feature representation with lower time-steps, enabling high performance and ultra-low latency in visual detection tasks. In this study, we conduct extensive evaluation of the tdIF method across two critical vision tasks: object detection and lane line detection. The results demonstrate that the proposed method surpasses current ANN-SNN conversion approaches, achieving state-of-the-art performance with ultra-low latency (within 5 time-steps). Our code is publicly available at https://github.com/zhangcj13/tdIF.

1. Introduction

Inspired by the brain's neuronal dynamics (Roy et al., 2019), Spiking Neural Networks (SNNs) (Maass, 1997) are anticipated to offer a low-power alternative to Artificial Neural Networks (ANNs), leveraging sparse, event-driven spikes for information transmission and processing. Over the past decade, ANNs have garnered significant attention across various industries and have been widely applied in fields such as computer vision, natural language processing, robotic control, etc. However, the increasing complexity of tasks and datasets necessitates more complex ANN architectures with larger parameter counts, which results in the increased demands for computational resources and energy consumption (Hubara et al., 2016). At this point, the low-energy consumption advantage of SNNs becomes evident as neurons in SNNs remain inactive and consume no energy until they receive a spike/event afferent (Christensen et al., 2022). From the perspective of computational hardware operations, multiplication operations in state-of-the-art deep ANNs are replaced by accumulate (AC) operations in SNNs, resulting in significantly lower energy consumption. In addition, SNNs enable a more power-efficient event-driven computing paradigm on neuromorphic hardware through the substitution of dense multiplication with sparse addition (Han et al., 2020). Meanwhile, as demonstrated in (Merolla et al., 2014; Davies et al., 2021; Painkras et al., 2013), SNNs implemented on designated neuromorphic processors can achieve energy consumption and latency that are several orders of magnitude lower than ANNs. Within the realm of neuromorphic computing, scholars are advancing neuromorphic computing architectures tailored for SNN applications, exemplified by systems such as TrueNorth (Merolla et al., 2014), Loihi (Davies et al., 2018) and Darwin3 (Ma et al., 2024). Additionally, numerous specialized hardware accelerators (Kim et al., 2023; Fang et al., 2024) tailored for SNNs have been implemented to optimally utilize spike-driven computation properties. These architectures, innovatively structured to mitigate the von Neumann bottleneck through the integration of memory and computation units, facilitate energy-efficient SNN inference processes.

Currently, two mainstream methodologies exist for developing supervised deep SNNs: direct training and conversion

[1]Zhejiang Key Laboratory of 3D Micro/Nano Fabrication and Characterization, Westlake Institute for Optoelectronics, Fuyang, Hangzhou 311421, China [2]CenBRAIN Neurotech, School of Engineering, Westlake University, Hangzhou, 310030, Zhejiang, China. Correspondence to: Jie Yang <yangjie@westlake.edu.cn>, Mohamad Sawan <sawan@westlake.edu.cn>.

from ANNs. Gradient-descent backpropagation (Rumelhart et al., 1986), which has been highly successful in ANNs, can not be directly applicable to SNNs due to the discontinuous functionality of spiking neurons. To address the non-differentiability issue of spiking functions, recent researchs have achieved substantial advancements in the direct training of SNNs through the application of backpropagation algorithms, facilitated by the utilization of surrogate gradients (Wu et al., 2018; Gu et al., 2019; Fang et al., 2021). Subsequently, they employ Backpropagation Through Time (BPTT) to optimize SNNs, which mirrors the backpropagation technique employed in conventional ANNs. However, computational and memory inefficiencies persist in this method, especially when implemented on common GPU platforms (Wu et al., 2021b), owing to the absence of tailored optimizations for binary event handling. Additionally, the sparsity of spike trains and the vanishing/exploding gradient problem further limit the effectiveness of direct training SNNs for high-complexity tasks.

In contrast, ANN-SNN conversion approaches, as an alternative solution, employ identical training paradigms as ANNs, thereby inheriting their computational efficiency advantages while requiring significantly fewer training resources than direct SNN training techniques (Cao et al., 2015; Diehl et al., 2015). The primary goal of ANN-SNN conversion is to align analog neuron activations with spiking neuron outputs, typically measured via firing rates (Diehl et al., 2015). Through emulating ANN activation patterns via SNN firing rates, the ANN-SNN conversion paradigm has achieved remarkable accuracy in challenging applications, attaining performance levels comparable to conventional ANNs (Hu et al., 2021; Kim et al., 2020b; Yan et al., 2021). However, all existing methods suffer from quantization and residual potential errors (as detailed later in Section 4), resulting in compromised performance metrics throughout the conversion procedure, particularly in cases of low-latency in visual detection tasks which require precise representations. While increasing inference latency mitigates these errors, it incurs higher computational costs and delays, undermining SNNs' efficiency advantages.

In this work, we implement a low-latency and high-performance ANN-SNN conversion method for visual detection tasks. Our main contributions are summarized as follows:

- Through mathematical analysis, we characterize the operational equivalence between ANN and SNN computational units, and focus on analyzing residual potential errors (RPE) and quantization errors (QE).

- We propose delay-spike strategy to minimize RPE and develop a dedicated inference pipeline to minimize the operational delays associated with delay-spike approaches in actual inference.

- We develop a tdIF (temporal-dependent Integrate-and-Fire Neuron) model that enables neurons to effectively utilize temporal information, thereby significantly reducing the latency of SNNs. Meanwhile, SNN models utilizing tdIF neurons demonstrate energy consumption levels within the same order of magnitude as those using IF neurons during inference.

- We validate the effectiveness of our algorithm on two visual detection tasks: object detection (at 5 time-steps, achieving mAP of 74.41% on PASCAL VOC, 55.73% on COCO) and lane line detection (at 5 time-steps, achieving Acc of 95.48% on Tusimple, F1 of 76.65% on CULane)

2. Related Work

We will review current research progress from two key perspectives: ANN-SNN conversion methods and SNN-based detection methods.

2.1. ANN-SNN Conversion

The seminal work on ANN-SNN conversion by (Cao et al., 2015) established the functional correspondence between rectified linear unit (ReLU) activation and integrate-and-fire (IF) neuronal dynamics. The weight scaling method based on maximum possible activations is proposed by Diehl et al. (Diehl et al., 2015) to address quantization errors, which are identified as the key impediment to lossless ANN-SNN conversion. Rueckauer et al. (Rueckauer et al., 2017) enhance the weight normalization technique by employing the $99.9th$ percentile of neuronal activations. Additionally, they introduce the "reset-by-subtraction" mechanism, also referred to as "soft-reset" mechanism, in order to mitigate the requirement for excessive time-steps. Meanwhile, Han et al. (Han et al., 2020) employ soft-reset spiking neurons with residual membrane potential (RMP) retention to better approximate ReLU functionality and mitigate ANN-SNN conversion loss. To eliminate conversion errors, many recent approaches (Deng & Gu, 2021; Ding et al., 2021) have adopted the strategy of replacing the activation functions in source ANN training. A Rate Normalization Layer is developed by Ding et al. (Ding et al., 2021) to replace ReLU activation functions in source ANN training, thereby enabling seamless conversion to SNNs. Bu et al. (Bu et al., 2023) introduce a novel quantization clip-floor-shift (QCFS) activation mechanism for ANN training, specifically designed to reduce conversion errors and enable high-accuracy SNN implementation with minimal time-steps. Jiang et al. (Jiang et al., 2023) present SlipReLU, a hybrid activation function combining threshold-ReLU and step function characteristics through weighted summation, effectively balancing the dual objectives of maintaining ANN accuracy while minimizing SNN conversion errors. However, these

methods focus exclusively on classification tasks and lack comprehensive experimental validation for visual detection tasks.

2.2. Detection based on SNNs

For detection tasks, recent advancements in ANN-SNN conversion have enabled the conversion of pre-trained ANN detectors into SNN implementations. Notably, Spiking-YOLO (Kim et al., 2020b) achieves performance comparable to conventional ANN detectors through specialized norm techniques, though demanding 8000 time-steps to match its ANN equivalent's accuracy. Farhadi et al. (Li et al., 2022) utilize Spike Calibration (SpikCalib) to detect SIN errors and rectify false spikes, enabling the accomplishment of various visual perception tasks, while still take 512 time-steps to achieve good performance. Wang et al.'s novel two-stage conversion algorithm (Wang et al., 2023) significantly minimized three critical error sources in ANN-SNN conversion: quantization errors, clipping errors, and residual membrane potential errors, while maintaining detection capability with 40 time-steps. Through optimizing the clipping threshold and designing layer-specific distribution patterns for synaptic weights and neuronal activations, Fast-SNN (Hu et al., 2023) accomplishes remarkable object detection accuracy with an exceptionally low-latency of 7 time-steps. Nevertheless, these low-latency ANN-SNN conversion methods still require large time-steps (Kim et al., 2020b; Li et al., 2022) or additional training costs after conversion (Wang et al., 2023; Hu et al., 2023) to get better performance on detection tasks.

Meanwhile, recent studies have explored the implementation of surrogate gradient methods (Wu et al., 2018) in detection tasks. Using surrogate gradient learning, Cordone et al. (Cordone et al., 2022) integrate a SNN into an SSD framework for object detection purposes. Su et al. (Su et al., 2023) introduce a pioneering framework, EMS-YOLO, enabling direct training of SNNs for object detection tasks and achieving competitive accuracy within merely 4 time-steps. Yuan et al. (Yuan et al., 2024) develop an end-to-end object detection SNN framework leveraging a surrogate gradient-enhanced BPTT algorithm, achieving competitive performance within 5 time-steps. Luo et al. (Luo et al., 2024) propose the SpikeYOLO framework integrating an I-LIF spiking neuron design, enabling integer-domain training protocols and event-triggered inference to push the performance limits of SNN. Direct training SNN methods, though capable of operating with reduced time-steps, are constrained by gradient instability (vanishing/exploding), resulting in inferior performance compared to ANN. In contrast to ANNs, the direct training of SNNs is notably more resource-intensive, requiring both increased memory allocation and roughly T times the computational duration.

For the above reasons, our approach prioritizes a one-stage ANN-SNN conversion framework for detection tasks. Our work shares similarities with the work presented in (Bu et al., 2023; Deng & Gu, 2021), which focuses on optimal conversion. However, our approach simultaneously reduces quantization and residual membrane potential errors, enabling visual detection at lower time-steps.

3. Preliminaries

Analog Neuron Model. In the context of ANNs, the computations performed by analog neurons can be streamlined as a concatenation of a linear transformation and a subsequent non-linear mapping. Specifically, analog neuron is formulated as detailed subsequently:

$$a^l = h(\mathbf{W}^l a^{l-1}) \tag{1}$$

where $l = 1, \ldots, L$ is the index of each layer, a^l is the output of the ReLU activation function $h(\cdot)$, and $\mathbf{W}^l$ represents the weight matrix connecting layer l to layer $l - 1$.

Spiking Neuron Model. IF neuron model, which is characterized by neuronal dynamics that enable spatiotemporal information processing and provides a robust representation of firing rate, is often employed in conversion algorithms for SNNs (Diehl et al., 2015; Rueckauer et al., 2017; Sengupta et al., 2019). The dynamics of IF neurons can be characterized:

$$M^l[t] = V^l[t-1] + \mathbf{W}^l S^{l-1}[t] \tag{2}$$
$$S^l[t] = \Theta(M^l[t] - \theta^l) \tag{3}$$
$$V^l[t] = M^l[t] - \theta^l S^l[t] \tag{4}$$

where t is the time-step of SNNs, $M^l[t]$ captures the sub-threshold membrane potential before spike initiation, while $V^l[t]$ reflects the potential after spike emission, θ^l is the firing threshold. The IF neuron generates spikes $S^l[t]$ with $\Theta(x)$ which represents the Heaviside step function, where $\Theta(x) = 1$ when $x >= 0$, and $\Theta(x) = 0$ when $x < 0$. The soft-reset (reset-by subtraction) mechanism (Han et al., 2020; Cao et al., 2015) which is presented as Eq. 4 is adopted to minimize the loss of information when the membrane potential surpasses the threshold, instead of resetting it to a constant value.

4. ANN-SNN Conversion Error

The objective of ANN-SNN conversion aims to create a consistent relationship where the spiking rate r^l in SNNs represents the analog activation a^l in ANNs. From the dynamics described in Eqs. 2~4, we can derive the functional dependence of membrane potential V^l on the input firing
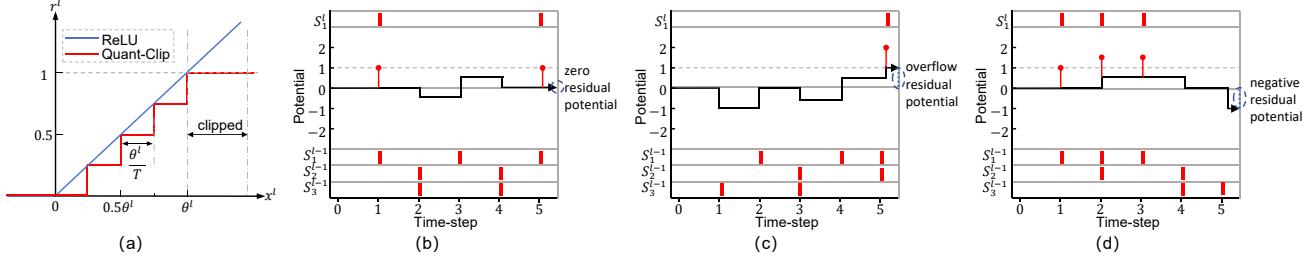
Figure 1. Conversion error between source ANN and converted SNN. (a) Errors between ReLU and Quant-Clip controlled by threshold θ^l and time-steps T. (b~d) Handcrafted examples of residual membrane potential representation error under the case of (b) zero residual potential, (c) overflow residual potential and (d) negative residual potential.

rate r^{l-1} as:

$$\frac{V^l[T] - V^l[0]}{T} = \mathbf{W}^l \sum_{t=1}^{T} \frac{S^{l-1}[t]}{T} - \theta^l \sum_{t=1}^{T} \frac{S^l[t]}{T} \quad (5)$$

where $\sum_{n=1}^{T} S^l[t]/T$ stand for the spiking rates r^l. Thus, we can reformulate the Eq. 5 as:

$$r^l = \frac{\mathbf{W}^l r^{l-1}}{\theta^l} - \epsilon^l \quad (6)$$

where ϵ^l equals $(V^l[T] - V^l[0])/(\theta^l \cdot T)$. Given equivalent input to both ANN and SNN at layer l, the output can be expressed as $x^l = \mathbf{W}^l r^{l-1}/\theta^l = \mathbf{W}^l a^{l-1}$. The absolute conversion error, which pertains to the comparison between the converted SNN and ANN at layer l, can be expressed as:

$$Err^l = r^l - a^l = x^l - \epsilon^l - h(x^l) \quad (7)$$

The conversion error Err^l is nonzero when $V^l[T] - V^l[0] \neq 0$ and $x^l > 0$. Firstly, we can employ a floor function to discretize the continuous activation values a^l into spiking rate representations r^l. Secondly, since ReLU-generated activations can be $a^l > 1$, whereas the spiking rate r^l is constrained within $\{0/T, 1/T, \cdots, T/T\}$, this discrepancy will result in clipping error as illustrated in Fig. 2. Here, a^l can be remapped to r^l by floor function and spike rate clipping technique, which is named as the quant-clip function:

$$r^l = clip\left(\frac{1}{T} \left\lfloor \frac{a^l T}{\lambda^l} \right\rfloor, 0, 1\right) \quad (8)$$

where $\lfloor \cdot \rfloor$ denotes the floor function, λ^l represents the maximum activation value ($max(a^l)$). Therefore, we can adopt Eq. 8 as the activation function for ANN to compensate for conversion error. However, there are still numerous errors that cannot be ignored.

It can be found the conversion error is mainly caused by ϵ^l. Here, we assume that the residual membrane potential falls within a level insufficient to trigger spike generation, as follows:

$$0 \le V^l[T] - V^l[0] < \theta^l \quad (9)$$

Under such assumption, the value of ϵ^l is restricted to the range $[0, 1/T)$, causing it to degenerate into the quantization error. However, the neuronal dynamics of IF neurons can lead to non-uniform spike distributions, resulting in residual membrane potentials that violate Eq. 9. To demonstrate this phenomenon, we present three representative cases in Fig. 1(b~d), where identical input firing rates produce different spike distribution patterns.

We suppose that the three spiking neurons in layer $l-1$ with $r^{l-1} = [0.6, 0.4, 0.4]$ in time-steps $T = 5$ are connected to one spiking neuron in layer l with $\mathbf{W}^l = [1, 0.5, -1]$ and firing threshold $\theta^l = 1$. Meanwhile, we also input this into the ANN to obtain the accurate output. According to Eq. 1, we can get $a^l = h([1, 0.5, -1] \cdot [0.6, 0.4, 0.4]^T) = 0.4$. As presented in Fig. 1(b), when the input spikes from the previous layer $l-1$ are relatively uniform, the firing rate is exactly $r^l = 2/5 = 0.4$ which equals a^l, and the residual membrane potential is precisely 0. However, as evidenced by Fig. 1(c~d), when the spike firing of layer $l-1$ is irregular, there will be an error between the output spike rate and a^l. Specifically, Fig. 1(c) shows a residual potential greater than 0, indicating incomplete spike firing which results in fewer spikes being emitted. Conversely, Fig. 1(d) depicts spikes being emitted prematurely and excessively, leading to a negative residual potential. Both scenarios reflect that errors in the firing rate will occur when the assumption condition Eq. 9 for the residual membrane potential does not hold.

The quantization error has unavoidable impact on various vision tasks. As shown in Fig. 1, the ReLU function of ANN can output continuous values, whereas the quant-clip function merely maps values in the range of $[0, \theta^l]$ onto discrete spike rates. However, for regression tasks, we believe that lower time-steps may fail to accurately represent some regression outputs. Here, we choose YOLO's anchor box mechanism (Redmon, 2016) as a case study for dimensional regression, where width and height predictions are computed through the following formula:

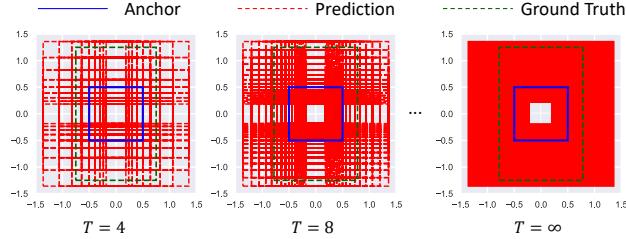$$w, h = exp(W^l r^{l-1}) \times anchor(w, h) \quad (10)$$

4

Figure 2. Illustration of inaccuracy in matching due to quantization errors at low time-steps in simplified bounding box regression based on YOLO anchor.

We assume that the predefined $anchor(w, h) = [1, 1]$ (the blue box in Fig. 2), and the output layer has two neurons with $\mathbf{W}^l = [1, -1]$. Under different quantized time-steps, the range that the final detection box can cover is illustrated in Fig. 2 with red color. Since the sizes of bounding boxes for detection tasks vary widely, at low time-steps, it inevitably leads to a mismatch between the prediction box and the ground truth, thereby reducing detection accuracy. For instance, at low time-steps $T = 4, 8$ in Fig. 2, the sizes of prediction boxes cannot accurately match our assumed ground truth (Green box in Fig. 2) perfectly. Meanwhile, in the experimental section 6, we also present data demonstrating the accuracy loss caused by quantization error at low quantized time-steps.

5. Methods

Based on the above analysis, we propose tdIF Neuron toward lossless conversion with low time-steps. Overall, in this chapter, we initiate our approach with a source neural network that employs the trainable quant-clip function as its activation function, which enables the transition of this network to its corresponding SNN version. The primary cause of the error in the SNN after applying the quant-clip function conversion is due to the residual potential and quantization error. Basically, we can reduce these errors by increasing the time-steps T of quant-clip function. However, following the conversion of the network to an SNN, increasing T not only diminishes the inference efficiency of SNN but also elevates power consumption. Therefore, secondly, we introduce the delay-spike rules in tdIF to reduce errors caused by irregular spiking and propose an inference pipeline to minimize the operational delays associated with delay-spike. Finally, we will present how our tdIF utilizes information from time-step to significantly reduce the time-steps in SNNs while also minimizing quantization errors.

5.1. ANN-SNN Conversion with Quant-Clip Function

The foundational premise of ANN-SNN conversion establishes a rate-coding equivalence between spiking neural dynamics and analog activation patterns. To minimize the

conversion error, we integrate quantization and clipping constraints into ANN training by employing modified activation functions (quant-clip). To distinguish it from the parameters in SNNs, the activation function $\overline{h}(\cdot)$ used in ANN training can be reformulated based on the quant-clip function Eq. 8 as follows:

$$\hat{a}^l = \overline{h}(x^l) = \lambda^l a^l = \lambda^l clip\left(\frac{1}{L}\left\lfloor \frac{x^l L}{\lambda^l} \right\rfloor, 0, 1\right) \quad (11)$$

where the hyperparameter L denotes quantized time-steps of ANNs. The maximum value λ^l of a^l in ANNs is designated as a trainable parameter, aiming to achieve a closer mapping relationship with the firing rate r^l in SNNs.

With this new activation function, we can prove that the estimated conversion error between SNNs and ANNs is zero. When an ANN with activation function Eq. 11 is converted to an SNN with the same weight. According to Eq. 7, the conversion error can be written as:

$$Err^l = r^l - a^l$$
$$= clip\left(\frac{1}{T}\left\lfloor \frac{x^l T}{\theta^l} \right\rfloor, 0, 1\right) - clip\left(\frac{1}{L}\left\lfloor \frac{x^l L}{\lambda^l} \right\rfloor, 0, 1\right) \quad (12)$$

If we have the conditions $T = L$, $\theta^l = \lambda^l$, then the conversion error $Err^l = 0$.

As shown in Fig. 3(b), when completing the source ANN training with quant-clip, we integrated Conv (convolution) and BN (batch-normalization) to eliminate unnecessary multiplication operations. Taking a single neuron as an example, we introduce the merging scheme of Conv and BN layers. Given an input of last layer $\hat{a}^l$, as the Conv layer precedes BN, a common architecture in ANNs, the output after Conv transformations can be expressed as:

$$y = \mathbf{W}^l \hat{a}^l, x^{l+1} = \gamma^l \frac{y - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta^l \quad (13)$$

where μ and σ^2 denote the mean and variance of data, respectively. γ^l and β^l are learnable affine parameters, and ϵ denotes a numerical stabilizer to ensure non-zero denominators. Meanwhile, in Eq. 11, parameter λ^l is introduced for adapting to the relationship between a^l and spike firing rates r^l. Crucially, both λ^l and λ^{l-1} of the previous activation function can be pre-computed and fused into Conv weights during inference, thereby preserving the binary (0/1) spike transmission characteristics in the converted SNN. All of these parameters are fixed during inference. The membrane potential at time-step t is therefore computed with $\hat{\mathbf{W}}^l S^l[t] + \hat{\mathbf{B}}^l$. The new parameters of the Conv layer can
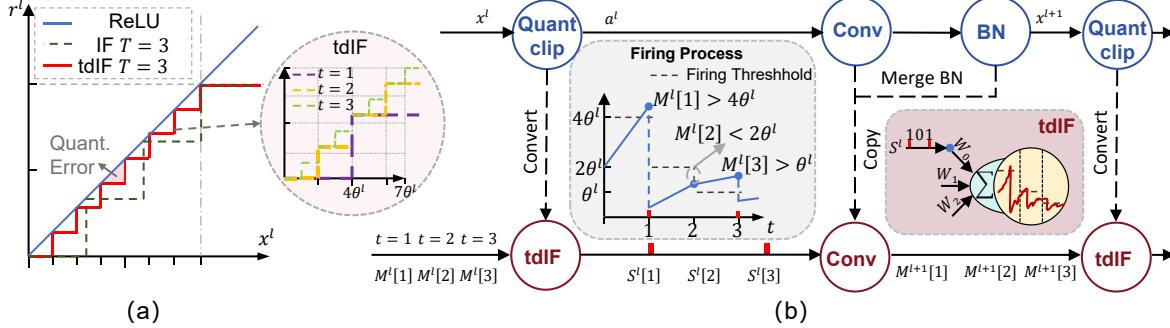
Figure 3. (a) Comparison between the firing rates of the IF neuron and the tdIF at the same time-steps setting ($T = 3$). By using time-steps to represent the positional information of binary numbers, our 3 time-steps tdIF achieves stronger information expressive capability equivalent to that of IF neurons within 7 time-steps. (b) Overall conversion framework and the dynamics of tdIF neuron.

be written as follows

$$\hat{\mathbf{W}}^l = \frac{\lambda^{l-1}\gamma^l}{\lambda^l\sqrt{\sigma^2 + \epsilon}}\mathbf{W}^l \tag{14}$$

$$\hat{\mathbf{B}}^l = -\frac{\mu\gamma^l}{\lambda^l\sqrt{\sigma^2 + \epsilon}} + \frac{\beta^l}{\lambda^l} \tag{15}$$

Subsequently, this weight configuration strategy effectively eliminates redundant multiplication operations in the SNN implementation.

5.2. Delay-Spike

Due to the irregular firing of spikes leading to residual potential error, we design a two-stage delay-spike algorithm to achieve accurate firing rate and prevent the occurrence of residual membrane potential. The algorithmic rule is shown in Alg. 1. At stage 1, we first accumulate the input value until the time t reaches time-delay T_{delay}, thereby yielding the current accumulated membrane potential $\overline{M}$ of neurons. Then, when time t reaches T_{delay}, the neuron begins to emit spikes. In the second stage, based on the delay step length, the remaining membrane potential will be directly utilized to calculate and emit the remaining spikes.

When the time-delay $T_{delay} = T$, the membrane potential $\overline{M}^l = \mathbf{W}^l \sum_{t=1}^T \frac{S^{l-1}[t]}{T}$. If we assume that the input to this layer is unbiased, according to Eq. 11, the current accumulated membrane potential is $\overline{M}^l = T \times x^l$. Given the conditions $\lambda^l = \theta^l$ and $0 \leq x^l < \theta^l$, then spikes are generated based on the $\overline{M}^l$ until it diminishes to a level where further spike generation is no longer feasible, therefore, it can be asserted that hypothesis Eq. 9 remains valid as :

$$0 \leq \overline{M}^l - \theta^l \sum_{t=1}^T S^l[t] < \theta^l \tag{16}$$

Under such hypothetical conditions, we recalculate the spike generation according to the firing pattern depicted in Fig.

Algorithm 1 Delay-Spike

input T_{delay}, $\overline{M}^l = V^l[0]$, SNNs infer mode is set as multi-step mode, the spike of previous layer $\{S^{l-1}[1], S^{l-1}[2], \cdots, S^{l-1}[T]\}$

output S^l

1: STAGE 1: Accumulating potential and delay spiking
2: **for** $t = 1, \cdots T$ **do**
3: $\quad \overline{M}^l = \overline{M}^l + \mathbf{W}^l S^{l-1}[t]$
4: $\quad$ **if** $t > T_{delay}$ **then**
5: $\quad\quad S^l[t - T_{delay}] = \Theta(\overline{\mathbf{M}}^l - \theta^l)$
6: $\quad\quad \overline{M}^l = \overline{M}^l - \theta^l S^l[t - T_{delay}]$
7: $\quad$ **end if**
8: **end for**
9: STAGE 2: Firing the remaining spikes
10: **for** $t = T - T_{delay}, \cdots T$ **do**
11: $\quad S^l[t] = \Theta(\overline{\mathbf{M}}^l - \theta^l), \overline{M}^l = \overline{M}^l - \theta^l S^l[t]$
12: **end for**

1(b~d) with this delay-spike method. The spike generation patterns for three cases are shown in Fig. 4. In this figure, during stage 1, the cumulative membrane potential at the final time-step remains consistent across all three cases, as the spike-firing rate of the preceding layer is identical, regardless of the specific firing patterns. Meanwhile, results in Fig. 4 (stage 2) confirm zero residual membrane potential after the final spiking event under our delay-spike method, which minimizes conversion errors caused by residual potential.

5.3. Pipelining of Delay-Spike

Similar to work (Liu et al., 2022), adopting a time-step-driven inference approach, SNNs based on the delay-spike strategy inevitably require extended inference time-steps to execute a single inference. Inspired by the allocation of different processing units to different layers to minimize memory access costs, we employ inter-layer direct transmis-
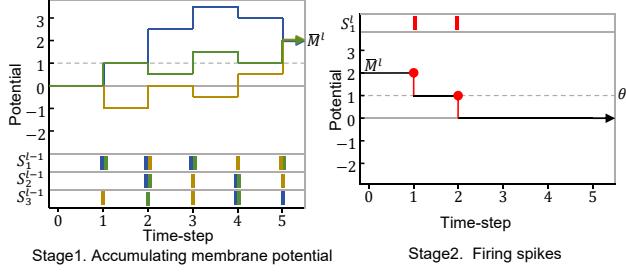
6

Figure 4. Example of delay-spike process under $T_{delay} = 5$.

sion and inter-sample pipelining to reduce the time latency of individual samples while maximizing the throughput of multiple samples.

Fig. 5 illustrates the inference pipeline for a 3-layer SNN with time-delay. The horizontal axis corresponds to the temporal dimension of inference (time-steps), with the vertical axis enumerating the network layers in sequential order. The diagram displays the distinct operational states of each layer's processing unit during membrane potential accumulation and spike firing. We set the duration of both phases (accumulation and firing) for each layer to the same value T. During time-step $t = [0, T]$, layer i begins spike firing synchronously at $t = T_{delay}$ after completing partial membrane potential accumulation. Its output is directly transmitted to layer $i + 1$ for simultaneous potential accumulation. Inter-layer direct transmission reduces the inference latency of individual samples. Based on this pipeline, for an SNN with n layers, the total inference time-steps are $T + n \times T_{delay}$. Additionally, sample 2 begins the accumulation phase in layer at time-step T to fully utilize the resources of layer i, thereby avoiding idle hardware waste. Notably, for multi-time-steps layer-driven inference methods, the delay-spike strategy does not increase the computational complexity of intra-neuron operations, and the on-chip inference time remains largely unaffected.

5.4. Temporal-Dependent IF Neuron

For current frequency-based conversion methods, the time-steps are often unrelated to each other in terms of sequence. The emission of spikes only affects the magnitude of the frequency, regardless of which time-step the spike is emitted in. To fully leverage the temporal information embedded within each time-step, we propose a neuron model with a dynamic threshold that adapts based on the time-step information. In this neuron, we adopt a binary coding approach, where time-steps are represented as binary bit information. Therefore, based on Eqs. 2∼4, the membrane potential accumulation rule and spike emission rule for neurons that utilize time-step information can be expressed as follows:
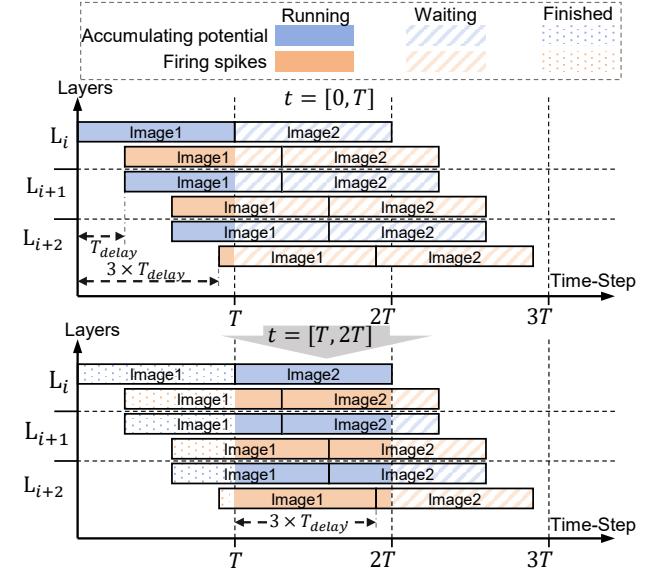

Figure 5. An example of the pipline for SNN with Delay-Spike.

$$M^l[t] = V^l[t-1] + c[t]\mathbf{W}^l S^{l-1}[t] \qquad (17)$$

$$S^l[t] = \Theta(M^l[t] - c[t]\theta^l) \qquad (18)$$

$$V^l[t] = M^l[t] - c[t]\theta^l S^l[t] \qquad (19)$$

Where $c[t] = 2^{T-t}$ represents the temporal information of spike binary coding at time-step t. Eq. 17 shows that the input membrane potential is multiplied by $c[t]$ according to the time-step. The spiking process Eq. 18 and the soft reset mechanism Eq. 19 utilize dynamic firing thresholds at different time-steps to calculate the operation of spike firing and membrane potential reset. The firing process is also illustrated in Fig. 3(b). Therefore, Eq. 5 can be modified as follows:

$$\frac{V^l[T] - V^l[0]}{T'} = \mathbf{W}^l \sum_{t=1}^{T} \frac{c[t]S^{l-1}[t]}{T'} - \theta^l \sum_{t=1}^{T} \frac{c[t]S^l[t]}{T'}$$
$$(20)$$

where $T' = \sum_{t=1}^{T} c[t] = 2^T - 1$. Hence, the $r^l = \sum_{n=1}^{T} c[t]S^l[t]/T'$ of the tdIF can also be mapped to a^l of the ANN.

Furthermore, according to the rules of binary encoding, our tdIF is capable of representing more information with fewer time-steps. A 3 time-steps tdIF neuron example, shown in Fig. 3(a), demonstrates how our tdIF combines temporal information to represent binary coding. Fig. 3(a) presents comparative activation curves demonstrating the input current x^l to firing-rate relationships for three neuron models: conventional ReLU, standard IF, and our proposed tdIF. As evident from this figure, our tdIF model can match information equivalent to approximately 7 time-steps within just 3

time-steps, effectively reducing the quantization error compared to ReLU. Meanwhile, within the same 3 time-steps, compared to IF neurons, our tdIF demonstrates superior expressive capability, enabling it to maintain high precision while ensuring low-latency characteristics.

5.5. Decoding of tdIF.

In regression tasks, the final layer of conventional ANNs typically generates numerical values through direct convolutional or fully-connected layers. When converted to SNNs, the final output corresponds to the membrane potential value defined as $V^o[t] = \mathbf{W}^o S^l[t]$, where $t \in [1, T]$, $\mathbf{W}^o$ is the weights of output layer and $S^l[t]$ is the input spikes. In implementations using IF neuron models, the equivalent output value relative to the source ANN is expressed becomes:

$$O_{IF} = \frac{\sum_{t=1}^{T} V^o[t]}{T} \tag{21}$$

When adopting the tdIF model, the temporal information embedded in discrete time-steps is encoded as positional indices in binary representations. Consequently, the final output membrane potential exhibits varying contributions from different time-step. The decoding methodology can be mathematically formulated as follows:

$$O_{tdIF} = \frac{\sum_{t=1}^{T} V^o[t] \times 2^{T-t}}{2^T - 1} \tag{22}$$

where O_{tdIF} denotes the decoded output, 2^{T-t} represents time-dependent weighting coefficients, and corresponds to the membrane potential value at time-step. Thus, the final output obtained through this method is equivalent to the original output of source ANN, and it can be applied as the final decoding component for various types of regression tasks.

6. Experiments

To demonstrate the effectiveness and efficiency of our algorithm for visual detection tasks, we compare our tdIF method with existing state-of-the-art approaches for image object detection and lane line detection tasks in this section.

We evaluate object detection performance across two benchmark datasets: PASCAL VOC (Everingham et al., 2010) and MS COCO 2017 (Wang et al., 2023). The baseline architecture adopts a streamlined YOLOv3-Tiny implementation (Farhadi & Redmon, 2018), modified for SNN compatibility through two key adaptations: substitution of leakyReLU activations with standard ReLU, and replacement of max pooling operations with average pooling. To assess scalability, we additionally implement a YOLOv3 variant with a backbone of ResNet34 (He et al., 2016). We compare our
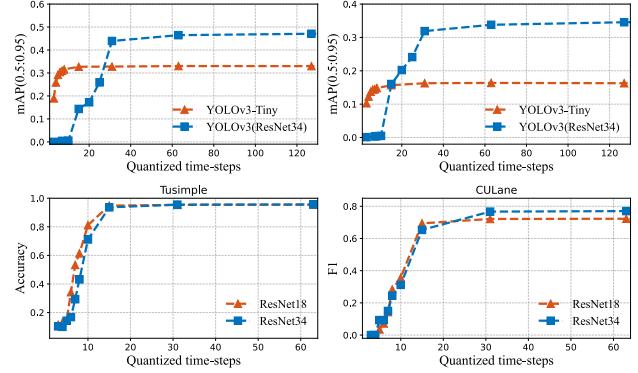

Figure 6. Accuracy gap between source ANN with Quant-Clip Function at different quant time-steps.

method with the contemporary ANN-SNN conversion methods (Kim et al., 2020b;a) and the Two-Stage Optimization method (Wang et al., 2023), Fast-SNN (Hu et al., 2023).

For lane line detection task, we also conducte experiments on two datasets: Tusimple (Pizzati et al., 2020) and CULane (Pan et al., 2018). We mirror the head structure from CondlaneNet (Liu et al., 2021) as our lane detection head. In Condlane-head, we adopt the membrane potential of the final output layer to represent the parameter map, which serves as the parameter for dynamic convolutional kernels. Meanwhile, the spike-rate is utilized to characterize the location map and offset map. Furthermore, our experiment incorporates two distinct ResNet configurations - ResNet18 and ResNet34 - as feature extraction networks, systematically assessing the depth adaptability of our conversion methodology in lane detection scenarios.

More information regarding the experimental setting can be found in Appendix .1.

Firstly, we investigate the accuracy loss due to quantization error at different time-steps. As shown in Fig. 6, we employ various quantization time-steps on four identical datasets. For all tasks across the datasets, when the time-steps is less than 8, the task accuracy is significantly lower than that of ANNs. When the time-step exceeds 32, the accuracy basically converges and approaches the values of ANNs. Fig. 6 also indicates that deep networks require relatively more time-steps to achieve accuracy convergence. This figure also demonstrates the accuracy loss due to quantization at low time-steps, as mentioned in sec. 4.

6.1. Performance

In this section, all results of our method employ the configuration of $T_{delay} = T$.

Table 1. Performance comparison for object detection task on PASCAL VOC and MS COCO.

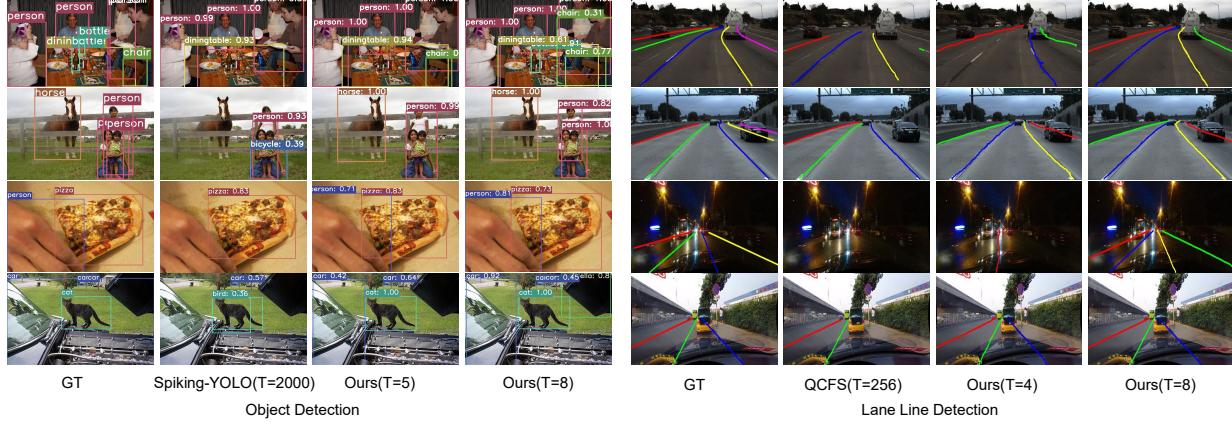| Method	Architecture	Param (M)	Time-steps	PASCAL VOC		MS COCO	
				AP@0.50:0.95	AP@0.50	AP@0.50:0.95	AP@0.50
Spiking-YOLO	YOLOv3-Tiny	8.85	8	0.00	0.23	0.00	0.11
(Kim et al., 2020b)	YOLOv3-Tiny	8.85	2000	26.9	59.9	12.51	25.78
Spike Calibration	YOLO(ResNet50)	45.24	64	31.75	63.85	16.17	33.12
(Li et al., 2022)	YOLO(ResNet50)	45.24	512	44.36	75.21	23.50	45.42
Two-Stage Optimization	YOLOV3-Tiny	8.85	20	28.5	64.1	-	-
(Wang et al., 2023)	YOLOV3-Tiny	8.85	40	30.7	66.4	-	-
Fast-SNN	Tiny YOLO	15.86	7	-	52.83	-	26.49
	Tiny YOLO	15.86	15	-	53.17	-	27.59
(Hu et al., 2023)	YOLOv2(ResNet34)	55.01	7	-	73.43	-	41.89
	YOLOv2(ResNet34)	55.01	15	-	76.05	-	46.40
Ours	YOLOv3-Tiny	8.85	5	32.33	63.96	16.26	36.12
	YOLOv3-Tiny	8.85	7	32.76	64.43	16.26	36.25
	YOLOv3-Tiny	8.85	8	32.80	64.53	16.30	36.29
	YOLOv3(ResNet34)	41.98	5	43.46	74.41	31.93	55.73
	YOLOv3(ResNet34)	41.98	7	**47.33**	**78.80**	34.50	58.89
	YOLOv3(ResNet34)	41.98	8	46.46	78.46	**35.16**	**59.78**

Figure 7. Visual quality comparison of object detection results with the network architecture YOLOv3(ResNet34) and lane line detection with ResNet34+Condlane.

6.1.1. OBJECT DETECTION

We summarize and compare the performance in Table 1. We include the Spiking-YOLO (Kim et al., 2020b), spike Calibration (Li et al., 2022), Two-Stage Optimization (Wang et al., 2023) and Fast-SNN (Hu et al., 2023) for comparison. These numbers are either taken from the corresponding papers or recalculated using their open-source code. Our algorithm achieves optimal detection accuracy in both datasets. Even with around 5 time-steps and similar model parameters, our method significantly outperforms others. Spiking-YOLO barely has accuracy at short time-steps, and even when other time-steps exceed 2000, it remains inferior to our YOLOv3-tiny model (5.4 percentage points lower on Pascal VOC and 3.35 percentage points lower on COCO under mean $AP@0.50:0.95$). Spike Calibration, despite having a notably superior feature extractor, still requires

up to 512 time-steps to match the accuracy of our model using ResNet34. The Two-Stage Optimization method also requires a higher number of time-steps, exceeding 40, to outperform us on VOC but is approximately 1.13 percentage points lower than our tiny model on mean $AP@0.50:0.95$. Fast-SNN has lower time-stepp requirements, but its method still falls below our algorithm. At the same time-steps of 7, our algorithm significantly outperforms Fast-SNN on both datasets. Additionally, Fast-SNN is a two-stage model that demands more computational resources. Meanwhile, our model experiences lower accuracy loss with shallower layers at different steps(with YOLOV3-Tiny, 0.47 percentage loss on VOC, 0.04 percentage loss on COCO), but higher loss with deeper layers (with YOLOV3-Resnet34, 3.0 on VOC, 3.23 on COCO).

Table 2. Performance comparison for Lane line detection task on Tusimple and CULane.

Method	Backbone	Time-steps	Tusimple			CULane		
			Acc(%) ↑	FP(%) ↓	FN(%) ↓	F1(%) ↑	Prec.(%) ↑	Recall(%) ↑
QCFS (Bu et al., 2023)	ResNet18	128	60.18	35.96	65.24	37.9	93.71	23.79
		256	92.25	2.25	9.03	61.33	90.95	46.26
	ResNet34	128	64.01	21.87	56.89	7.79	83.83	4.09
		256	93.88	3.29	6.82	48.35	**94.17**	32.53
Ours	ResNet18	4	94.80	3.59	5.29	69.32	80.67	60.77
		5	95.35	2.53	4.27	72.17	80.30	65.53
		7	95.37	2.41	4.31	72.23	79.82	65.97
		8	95.29	2.63	4.36	72.35	81.15	65.27
	ResNet34	4	93.55	6.02	7.89	65.38	78.80	55.87
		5	95.48	2.53	4.14	76.65	81.29	72.52
		7	95.53	**2.33**	**3.98**	77.17	81.93	72.93
		8	**95.56**	2.47	4.13	**77.61**	82.20	**73.51**

6.1.2. LANE LINE DETECTION

We summarize and compare the performance in Table 2. Since we are the first to adopt the ANN-SNN conversion algorithm for lane line detection task, we apply the QCFS method to lane detection tasks for comparison. Both methods use CondLane as the detection head and refer to Cond-Lane's training strategies for weight optimization. Our algorithm achieves high detection accuracy within a short number of time-steps, reaching over 94% on the TuSimple dataset with 4 time-steps and over 69% on the CULane dataset. Experimental results demonstrate that the model achieves consistent accuracy when the time-steps extends beyond 5. However, due to poor handling of cumulative errors, OCFS has almost poor performance when the number of time-steps is less than 128 and requires a larger number of time-steps (greater than 256) to achieve corresponding detection accuracy. Given the limited number of samples in the TuSimple dataset (around 3000 in total), using ResNet18 and ResNet34 yields similar performance, with a difference in detection accuracy of only 0.27% at 8 time-steps. In contrast, the difference in performance between these two networks on the CULane dataset is around 5.26%. Additionally, deeper networks experience greater losses at low latency, with ResNet34 showing a difference in accuracy of 2.01% between 8 and 4 time-steps, while ResNet18 only shows a difference of 0.49% on the TuSimple dataset.

In Fig. 7, we further provide visual results based on ResNet34, encompassing both object detection and lane detection. As illustrated, our SNN is capable of producing detection results close to the ground truth with lower time-steps. For instance, our algorithm with a latency of 8 can differentiate some overlapping objects (e.g., bottle and person) and precise lane lines with higher accuracy. Appendix .2 contains more information on the detection accuracy of different quantized time-steps with time-steps

$[3, 4, 5, 7, 8, 10]$ across four datasets.

6.2. Ablation Studies

Figure 8. Ablation study with/without tdIF and different time-delay.

We conduct ablation experiments combining different neuron types (IF and tdIF) and different time-delay on Pascal VOC dataset and Tusimple dataset at an 8 time-steps setup. Fig. 8 presents the results of our experiments. The x-axis represents different time-delay ranges from 0 to 8. It can be seen that the combination of tdIF and delay-spike can effectively reduce the time-steps while achieving a higher detection accuracy. Meanwhile, equipped with tdIF relies more heavily on the time-delay term (our tdIF has no accuracy when $T_{delay} = 0, 1$). We attribute this to tdIF's explicit incorporation of temporal information (adjust membrane

Table 3. Component of IF neuron and tdIF neuron on digital neuromorphic processor.

IF neuron	tdIF neuron	Description
Component for Accumulating potential.		
MLD(R0, ADD1, 1)	MLD(R0, ADD1, 1)	load weight W^l
MLD(R1, ADD2, 0)	MLD(R1, ADD2, 0)	load state $V^l[t-1]$
	SHL(R0, R0, R2)	$c[t] \times w_{ij}$ via bitwise shifting, R2=$T-t$
ADD(R1, R0, R1)	ADD(R1, R0, R1)	$M^l[t] = V^l[t-1] + W^l$
MST(ADD2, R1, 1)	MST(ADD2, R1, 1)	store R1 in $M^l[t]$
12.7	13.9	Energy for Accumulating potential (pJ)
Component for Firing spikes.		
MLD(R0, ADD2, 0)	MLD (R0, ADD2, 0)	load state $M^l[t]$
GTH(R2, R0, R1)	GTH(R2, R0, R1)	generate spike, IF:$\Theta(M^l[t] - \theta^l)$, tdIF:$\Theta(M^l[t] - \hat{\theta}^l[t])$
MUL(R3, R2, R1)	MUL(R3, R2, R1)	IF:$\theta \times S^l[t]$, tdIF:$\hat{\theta}[t] \times S^l[t]$
SUB(R0, R0, R3)	SUB(R0, R0, R3)	reset state if spike
MST(ADD2, R0, 1)	MST(ADD2, R0, 1)	store R0 in $V^l[t]$
EVC(R2)	EVC(R2)	capture event
13.2	13.2	Energy for Firing spikes (pJ)

Table 4. **Neuron Processing Energy Consumption.**

Instruction	Description	Energy (pJ)
ADD/SUB	Arithmetic ops.	1.4
GTH/MAX/MIN	Compare ops.	1.2
SHL/SHR	Bit-wise ops.	1.2
EVC	Event Capture	0.5
	+ if generates event	+ 1.1
MLD	Data Mem Load/Store	3.7
MST		3.9

potential accumulation based on time-steps). In Eq. 17, the first time-step exerts the strongest influence on membrane potential due to $c[1] = 2^{T-1}$. Consequently, spikes generated with temporal deviation (either delayed or mistimed) significantly disrupt the membrane potential accumulation. This temporal sensitivity makes our method more dependent on precise delay-spike coordination compared to conventional frequency-based IF strategies.

However, when employing deep network architectures with IF neurons, such as ResNet, they also fail to exhibit performance at $T_{delay} = 0$. As the amplified accumulation characteristic of tdIF enables faster membrane potential buildup during initial time-steps, our tdIF demonstrates significantly superior performance compared to models that utilize IF Neuron at $T_{delay} \geq 2$ and basically achieves peak performance at $T_{delay} = 3$, whereas IF requires substantially more delayed time to reach its highest levels. Without the adoption of tdIF, all models except for the shallow one like YOLOv3-Tiny exhibit significant losses compared to those with tdIF. Specifically, when T_{delay} is set to 8, there is a precision loss exceeding 10 percent.

6.3. Computation Cost

6.3.1. ENERGY CONSUMPTION FOR TDIF

Since our tdIF neuron introduces a multiplicative operation on the input membrane potential, it may incur additional energy consumption. Here, we first analyze the power consumption calculation of both IF and tdIF neurons. As mentioned in (Tang et al., 2023), we have the energy consumption of the digital neuromorphic processor for each operation in Table 4. Therefore, based on the operation counts of the digital neuromorphic processor, we can divide the membrane potential accumulation and spike generation of the IF neuron into two parts as shown in Table 3. Due to the binary (0,1) nature of spikes, the membrane potential accumulation process is spike-driven, while spike generation is computed at every time-step. This yields energy consumption values of $12.7pJ$ and $13.2pJ$ for these two processes in a single IF neuron, respectively.

Although the tdIF neuron incorporates a multiplication operation in its membrane potential accumulation (Eq. 17), it maintains the sparse (0,1) transmission characteristics between layers and preserves the event-driven computation property. Consequently, only addition operations are required when computing input membrane potentials in convolutional or fully-connected layers. For the amplification coefficient $c[t] = 2^{T-t}$ in Eq. 17, we can implement it using bit-shift operations. Thus, during the potential accumulation process shown in Table 3, tdIF requires only one additional left-shift operation compared to IF neuron. Furthermore, for the spike generation computation, we can store the threshold values per time-step as $\hat{\theta}^l = [2^{T-1}\theta^l, 2^{T-2}\theta^l, \cdots, \theta^l]$. This implementation results in energy consumption values of $13.9pJ$ and $13.2pJ$ for the two processes in tdIF neuron, representing only a $1.2pJ$ overall increase compared to the standard IF neuron.

Table 5. **Inference Energy Consumption**

Architecture	Time-steps	Neuron	Performance	Average Spiking rate	Power (mJ)	E/E_{ref}
			Results on Pascal VOC			
YOLOv3-Tiny	8	IF	61.60	0.079	12.1	1.0× (reference)
	32	IF	64.07	0.078	47.4	3.9×
	8	tdIF	64.53	0.173	31.0	2.6×
YOLOv3 (ResNet34)	8	IF	60.20	0.056	123.2	1.0× (reference)
	32	IF	74.40	0.056	475.6	3.8×
	8	tdIF	78.46	0.169	423.8	3.4×
			Results on Tusimple			
ResNet18 +Condlane	8	IF	67.75	0.060	113.4	1.0× (reference)
	16	IF	94.87	0.052	220.8	1.9×
	8	tdIF	95.27	0.165	271.3	2.4×
ResNet34 +Condlane	8	IF	44.65	0.041	118.0	1.0× (reference)
	16	IF	93.71	0.042	265.8	2.3×
	8	tdIF	95.56	0.152	387.5	3.3×

6.3.2. INFERENCE ENERGY COST

We compared the energy consumption during model inference between our approach and models using IF neurons across two detection tasks. As shown in Table 5, all models in this experiments adopted the $T_{delay} = T$ setting with 8 time-steps. To ensure fair comparison under similar performance levels, we configured the IF neuron models with 32 time-steps for object detection and 16 time-steps for lane detection task.

The models employing tdIF neurons consistently demonstrated an average firing rate of approximately 0.15 across all four architectures, significantly higher than the 0.06 firing rate of conventional IF neurons. Consequently, under identical time-step settings, tdIF-based models consume about 2 to 3 times more energy than their IF counterparts. However, to achieve comparable performance to our tdIF models, IF-based models require increased time-steps, inevitably leading to higher power consumption. For instance, the YOLOv3 model on Pascal VOC dataset must extend to 32 time-steps, resulting in substantially higher power consumption than our approach. On the Tusimple dataset, our tdIF model's power consumption is only about 40% higher than the 16 time-steps IF-based model. These results demonstrate that while tdIF models increase energy consumption during neuronal integration and exhibit higher spike firing rates, their overall power consumption remains within the same order of magnitude as IF models while delivering superior performance.

7. Conclusion

In this work, we explore the potential of Spiking Neural Networks in visual detection tasks. This work elucidates the impact of residual membrane potential errors on rate-based method, alongside the influence of quantization errors on the accuracy of regression vision tasks. Addressing the challenges posed by these errors, our idea is simple and intuitive. To this end, we employ delay-spike strategy to mitigate residual potential errors arising from irregular spiking, and utilize temporal-dependent Integrate-and-Fire neuron architecture to enhance the feature representation capability of neural networks. The extensive experimental validation confirms that our proposed method achieves the state-of-the-art performance metrics across multiple vision tasks, delivering superior accuracy while maintaining low latency in both object detection and lane detection. Furthermore, our proposed neuron model demonstrates power consumption on neuromorphic hardware that remains within the same order of magnitude as conventional IF neurons. In summary, this research represents a substantial leap forward in deploying SNNs for visual detection applications, establishing a foundational framework for subsequent investigations in neuromorphic vision systems.

8. Impact Statement

Our paper proposes the delay-spike mechanism and the temporal-dependent Integrate-and-Fire (tdIF) neuron model, achieving low-latency performance in visual regression tasks. For any hardware device implementing layer-by-layer computation in spiking neural networks (SNNs), the delay-spike mechanism significantly reduces errors caused by residual membrane potential after conversion, without increasing computational complexity. Furthermore, the tdIF model demonstrates a remarkable reduction in latency for regression tasks, suggesting that SNNs can achieve excellent feature representation capabilities even with low time-steps, provided they fully utilize temporal information. This breakthrough highlights the potential of SNNs for ultra-low-power applications, paving the way for their widespread adoption in real-time, energy-efficient AI systems.

References

Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

Bu, T., Fang, W., Ding, J., Dai, P., Yu, Z., and Huang, T. Optimal ann-snn conversion for high-accuracy and ultra-low-latency spiking neural networks. *arXiv preprint arXiv:2303.04347*, 2023.

Cai, Z., Ravichandran, A., Maji, S., Fowlkes, C., Tu, Z., and Soatto, S. Exponential moving average normalization for self-supervised and semi-supervised learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 194–203, 2021.

Cao, Y., Chen, Y., and Khosla, D. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113: 54–66, 2015.

Christensen, D. V., Dittmann, R., Linares-Barranco, B., Sebastian, A., Le Gallo, M., Redaelli, A., Slesazeck, S., Mikolajick, T., Spiga, S., Menzel, S., et al. 2022 roadmap on neuromorphic computing and engineering. *Neuromorphic Computing and Engineering*, 2(2):022501, 2022.

Cordone, L., Miramond, B., and Thierion, P. Object detection with spiking neural networks on automotive event data. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2022.

Davies, M., Srinivasa, N., Lin, T.-H., Chinya, G., Cao, Y., Choday, S. H., Dimou, G., Joshi, P., Imam, N., Jain, S., et al. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 38(1):82–99, 2018.

Davies, M., Wild, A., Orchard, G., Sandamirskaya, Y., Guerra, G. A. F., Joshi, P., Plank, P., and Risbud, S. R. Advancing neuromorphic computing with loihi: A survey of results and outlook. *Proceedings of the IEEE*, 109(5): 911–934, 2021.

Deng, S. and Gu, S. Optimal conversion of conventional artificial neural networks to spiking neural networks. *arXiv preprint arXiv:2103.00476*, 2021.

Diehl, P. U., Neil, D., Binas, J., Cook, M., Liu, S.-C., and Pfeiffer, M. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 International joint conference on neural networks (IJCNN)*, pp. 1–8. ieee, 2015.

Ding, J., Yu, Z., Tian, Y., and Huang, T. Optimal ann-snn conversion for fast and accurate inference in deep spiking neural networks. *arXiv preprint arXiv:2105.11654*, 2021.

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88: 303–338, 2010.

Fang, C., Shen, Z., Wang, Z., Wang, C., Zhao, S., Tian, F., Yang, J., and Sawan, M. An energy-efficient unstructured sparsity-aware deep snn accelerator with 3-d computation array. *IEEE Journal of Solid-State Circuits*, 2024.

Fang, W., Yu, Z., Chen, Y., Masquelier, T., Huang, T., and Tian, Y. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 2661–2671, 2021.

Farhadi, A. and Redmon, J. Yolov3: An incremental improvement. In *Computer vision and pattern recognition*, volume 1804, pp. 1–6. Springer Berlin/Heidelberg, Germany, 2018.

Gu, P., Xiao, R., Pan, G., and Tang, H. Stca: Spatio-temporal credit assignment with delayed feedback in deep spiking neural networks. In *IJCAI*, volume 15, pp. 1366–1372, 2019.

Han, B., Srinivasan, G., and Roy, K. Rmp-snn: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 13558–13567, 2020.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hu, Y., Tang, H., and Pan, G. Spiking deep residual networks. *IEEE Transactions on Neural Networks and Learning Systems*, 34(8):5200–5205, 2021.

Hu, Y., Zheng, Q., Jiang, X., and Pan, G. Fast-snn: fast spiking neural network by converting quantized ann. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. Binarized neural networks. *Advances in neural information processing systems*, 29, 2016.

Jiang, H., Anumasa, S., De Masi, G., Xiong, H., and Gu, B. A unified optimization framework of ann-snn conversion: towards optimal mapping from activation values to firing rates. In *International Conference on Machine Learning*, pp. 14945–14974. PMLR, 2023.

Kim, S., Park, S., Na, B., Kim, J., and Yoon, S. Towards fast and accurate object detection in bio-inspired spiking

neural networks through bayesian optimization. *IEEE Access*, 9:2633–2643, 2020a.

Kim, S., Park, S., Na, B., and Yoon, S. Spiking-yolo: spiking neural network for energy-efficient object detection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 11270–11277, 2020b.

Kim, S., Kim, S., Hong, S., Kim, S., Han, D., and Yoo, H. C-dnn: A 24.5-85.8tops/w complementary-deep-neural-network processor with heterogeneous cnn/snn core architecture and forward-gradient-based sparsity generation. *2023 IEEE International Solid- State Circuits Conference (ISSCC)*, pp. 334–336, 2023.

Lee, J. H., Delbruck, T., and Pfeiffer, M. Training deep spiking neural networks using backpropagation. *Frontiers in neuroscience*, 10:508, 2016.

Li, Y. and Zeng, Y. Efficient and accurate conversion of spiking neural network with burst spikes. *arXiv preprint arXiv:2204.13271*, 2022.

Li, Y., Deng, S., Dong, X., Gong, R., and Gu, S. A free lunch from ann: Towards efficient, accurate spiking neural networks calibration. In *International conference on machine learning*, pp. 6316–6325. PMLR, 2021.

Li, Y., He, X., Dong, Y., Kong, Q., and Zeng, Y. Spike calibration: Fast and accurate conversion of spiking neural network for object detection and segmentation. *arXiv preprint arXiv:2207.02702*, 2022.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pp. 740–755. Springer, 2014.

Liu, F., Zhao, W., Chen, Y., Wang, Z., and Jiang, L. Spikeconverter: An efficient conversion framework zipping the gap between artificial neural networks and spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 1692–1701, 2022.

Liu, L., Chen, X., Zhu, S., and Tan, P. Condlanenet: a top-to-down lane detection framework based on conditional convolution. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3773–3782, 2021.

Luo, X., Yao, M., Chou, Y., Xu, B., and Li, G. Integer-valued training and spike-driven inference spiking neural network for high-performance and energy-efficient object detection. In *European Conference on Computer Vision*, pp. 253–272. Springer, 2024.

Lv, L., Fang, W., Yuan, L., and Tian, Y. Optimal ann-snn conversion with group neurons. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6475–6479. IEEE, 2024.

Ma, D., Jin, X., Sun, S., Li, Y., Wu, X., Hu, Y., Yang, F., Tang, H., Zhu, X., Lin, P., et al. Darwin3: a large-scale neuromorphic chip with a novel isa and on-chip learning. *National Science Review*, 11(5):nwae102, 2024.

Maass, W. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10(9): 1659–1671, 1997.

Merolla, P. A., Arthur, J. V., Alvarez-Icaza, R., Cassidy, A. S., Sawada, J., Akopyan, F., Jackson, B. L., Imam, N., Guo, C., Nakamura, Y., et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

Neftci, E. O., Mostafa, H., and Zenke, F. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

Painkras, E., Plana, L. A., Garside, J., Temple, S., Galluppi, F., Patterson, C., Lester, D. R., Brown, A. D., and Furber, S. B. Spinnaker: A 1-w 18-core system-on-chip for massively-parallel neural network simulation. *IEEE Journal of Solid-State Circuits*, 48(8):1943–1953, 2013.

Pan, X., Shi, J., Luo, P., Wang, X., and Tang, X. Spatial as deep: Spatial cnn for traffic scene understanding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Pizzati, F., Allodi, M., Barrera, A., and García, F. Lane detection and classification using cascaded cnns. In *Computer Aided Systems Theory–EUROCAST 2019: 17th International Conference, Las Palmas de Gran Canaria, Spain, February 17–22, 2019, Revised Selected Papers, Part II 17*, pp. 95–103. Springer, 2020.

Redmon, J. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

Roy, K., Jaiswal, A., and Panda, P. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, 2019.

Rueckauer, B., Lungu, I.-A., Hu, Y., Pfeiffer, M., and Liu, S.-C. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience*, 11:682, 2017.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Sengupta, A., Ye, Y., Wang, R., Liu, C., and Roy, K. Going deeper in spiking neural networks: Vgg and residual architectures. *Frontiers in neuroscience*, 13:95, 2019.

Su, Q., Chou, Y., Hu, Y., Li, J., Mei, S., Zhang, Z., and Li, G. Deep directly-trained spiking neural networks for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6555–6565, 2023.

Tang, G., Safa, A., Shidqi, K., Detterer, P., Traferro, S., Konijnenburg, M., Sifalakis, M., van Schaik, G.-J., and Yousefzadeh, A. Open the box of digital neuromorphic processor: Towards effective algorithm-hardware co-design. In *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5. IEEE, 2023.

Wang, Z., Zhang, Y., Lian, S., Cui, X., Yan, R., and Tang, H. Toward high-accuracy and low-latency spiking neural networks with two-stage optimization. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

Wu, J., Chua, Y., Zhang, M., Li, G., Li, H., and Tan, K. C. A tandem learning rule for effective training and rapid inference of deep spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 34 (1):446–460, 2021a.

Wu, J., Xu, C., Han, X., Zhou, D., Zhang, M., Li, H., and Tan, K. C. Progressive tandem learning for pattern recognition with deep spiking neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44 (11):7824–7840, 2021b.

Wu, Y., Deng, L., Li, G., Zhu, J., and Shi, L. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12: 331, 2018.

Yan, Z., Zhou, J., and Wong, W.-F. Near lossless transfer learning for spiking neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 10577–10584, 2021.

Yuan, M., Zhang, C., Wang, Z., Liu, H., Pan, G., and Tang, H. Trainable spiking-yolo for low-latency and high-performance object detection. *Neural Networks*, 172:106092, 2024.

Zheng, G., Songtao, L., Feng, W., Zeming, L., Jian, S., et al. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021a.

Zheng, H., Wu, Y., Deng, L., Hu, Y., and Li, G. Going deeper with directly-trained larger spiking neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11062–11070, 2021b.

The experimental results in this paper are reproducible. We explain the details of model training and configuration in the main text and supplement it in the appendix.

.1. Experimental Setting.

The implementation is in Pytorch and all the experiments are computed on a machine with an NVIDIA A800 GPU.

.1.1. PASCAL VOC AND MS COCO

The Pascal VOC dataset (Everingham et al., 2010) contains 16551 training images and 4952 validation images. For training preprocessing, we employ a range of multiscale image sizes to randomly resize the images. The actual multiscale ranges is $[416 - r \times 32, 416 + r \times 32]$, where $r \in [0, 5]$. In data augmentation, the strategies of Mosaic and MixUp from (Bochkovskiy et al., 2020) are utilized to enhance the diversity and robustness of training datasets. In MixUp, four different images are selected and placed in a specific manner (in a 2×2 grid) to form a new large image. We use $\alpha \times image_1 + (1 - \alpha) \times image_2$,where $\alpha = 0.5$, to generate our mixing image data. Additionally, We employ Exponential moving average (EMA) technology (Cai et al., 2021) in our training process. The learning rate is set to 0.01 and followed by a cosine decay schedule same as (Zheng et al., 2021a). The Weight decay is set to 5×10^{-4}, and the networks are optimized for 90 epochs. For test images, they are padded and resized to the same size. For all architectures we tested, the Max Pooling layers are replaced to Average Pooling layers. The input features of the YOLOv3 head are adjusted according to the number of channels in the output feature maps of ResNet34.

The MS COCO dataset (Wang et al., 2023) contains $118k$ training images and $5k$ validation images. The training technique remains consistent with our approach for training on VOC, but with modifications to the input range $[640 - r \times 32, 640 + r \times 32]$.

.1.2. TUSIMPLE AND CULANE

Tusimple (Pizzati et al., 2020) is a dataset of highway driving scenes, which contains $3.3k$ training images and $2.8k$ validation images. CULane(Pan et al., 2018). is a widely used large lane detection dataset with 9 different scenarios (Urban&highway), which contains $88.9k$ training images and $34.7k$ validation images.

For training preprocessing, we utilize techniques such as Random Blur, Brightness Adjustment, Rotation, and Resizing to augment the images. In the optimizing process, we use Adam optimizer and step learning rate decay with an initial learning rate of $3e - 4$. For each dataset, we train on the training set without any extra data. We respectively train 20 and 16 epochs for CULane and TuSimple with a batchsize of 128.

For CULane, we utilizes the F1 measure as in (Liu et al., 2021). IoU between the predicted lane line and GT label is taken for judging whether a sample is true positive (TP) or false positive (FP) or false negative (FN). IoU of two lines is defined as the IoU of their masks with a fixed line width. F1-measure is calculated as follows:

$$F1 = \frac{2 \times Prec \times Recall}{Prec + Recall}, \tag{23}$$

$$Prec = \frac{TP}{TP + FP} \tag{24}$$

$$Recall = \frac{TP}{TP + FN} \tag{25}$$

For TuSimple dataset, there are three official indicators: false-positive rate (FPR), false-negative rate (FNR), and accuracy.

$$Acc = \frac{\sum_{clip} C_{clip}}{\sum_{clip} S_{clip}} \tag{26}$$

Where C_{clip} is the number of correctly predicted lane points and S_{clip} is the total number of lane points of a clip.

.2. Additional results.

Our method employs various quantized time-steps to train the model and obtain the corresponding Spiking Neural Network performance (the same time-steps) based on these model weights. Here, we present the progress of conversion under different training models in Table 6, serving as supplementary information for further experimental results.

Table 6. **The trade-off between accuracy of the dataset and inference latency.**

Architecture	Quantized time-steps	ANN metric	time-steps					
			3	4	5	7	8	10
mAP@0.50 Results on Pascal VOC								
YOLOv3-Tiny	32	64.30	60.28	63.62	63.96	64.17	64.08	64.09
	64	64.74	60.24	63.72	64.43	64.80	64.80	64.81
	128	64.43	59.83	63.71	64.46	64.47	64.61	64.54
	256	64.57	60.34	63.77	64.19	64.60	64.56	64.81
YOLOv3 (ResNet34)	32	74.44	54.01	72.46	74.42	74.80	74.64	74.79
	64	77.89	25.90	71.62	77.14	78.10	77.89	77.96
	128	79.01	11.49	66.72	77.46	79.01	78.95	78.86
	256	78.48	3.73	53.16	76.12	78.39	78.55	78.79
mAP@0.50 Results on MS COCO								
YOLOv3-Tiny	32	36.17	32.52	35.67	36.12	36.28	36.31	36.19
	64	36.40	31.84	35.46	36.23	36.48	36.52	36.52
	128	36.27	31.40	35.53	35.88	36.25	36.27	36.22
	256	36.31	31.48	35.43	36.00	36.22	36.29	36.30
YOLOv3 (ResNet34)	32	56.21	41.95	53.70	55.73	56.02	56.03	55.98
	64	59.21	25.79	52.50	57.08	58.24	58.06	58.19
	128	60.51	11.90	50.78	57.50	58.89	58.82	58.85
	256	60.11	9.53	49.58	58.21	59.70	59.78	59.77
Acc(%) Results on Tusimple								
ResNet18 +Condlane	16	94.82	45.99	94.80	94.79	94.71	94.78	94.77
	32	95.34	19.85	89.21	95.35	95.40	95.39	95.42
	64	95.48	12.08	65.47	95.00	95.52	95.54	95.53
	128	95.38	10.62	58.53	94.65	95.37	95.39	95.37
ResNet34 +Condlane	16	93.62	14.26	93.55	87.45	69.67	68.54	68.32
	32	95.53	0.45	85.83	95.48	95.71	95.72	95.71
	64	95.65	0.41	61.82	95.31	95.68	95.71	95.70
	128	95.53	0.79	54.90	94.50	95.53	95.54	95.58
F1(%) Results on CULane								
ResNet18 +Condlane	16	69.41	35.69	69.32	69.80	70.03	70.04	70.03
	32	72.13	26.90	65.01	72.18	72.81	72.81	72.91
	64	72.26	8.00	59.65	70.84	72.60	72.60	72.62
	128	72.22	3.26	48.61	68.90	72.24	72.29	72.30
ResNet34 +Condlane	16	65.33	8.73	65.37	63.34	58.21	58.29	58.12
	32	76.68	3.45	67.33	76.64	77.73	77.74	77.77
	64	77.02	0.53	64.05	75.89	77.25	77.39	77.38
	128	77.59	0.12	49.86	74.28	77.16	77.27	77.31